How to Contribute to Mnemonic

Apache Mnemonic project developed by Java and C/C++ languages. Like any other Apache open source project, we have several ways to contribute to the project. Some of which are documentation improvements, code reviews, code changes, helping users on mailing lists, helping in release candidate testing, etc.

To become an effective contributor, we can follow certain guidelines for each of these contribution catogeries mentioned above. Let's go through the following sections for more details.

Documentation Changes

For any new contributor to the project, we strongly encourage to go through the documentation for better understanding the project. We can find the documentation from the Mnemonic website.

Note: this site is under construction at this stage, soon it will be active.

While going through the documentation, you may find some improvements/suggestion to the documentation. When you find these changes, please raise a JIRA ticket and if you want to fix it your won, we welcome you to create a pull request against trunk and submit it to JIRA.

Please look at the documentation for how to build website and documentation.[TODO: link to the page for building website and documentation page]

Code Reviews

Apache Mnemonic project code contributions will be submitted via pull requests and these pull request link will be commented in JIRA. Any one can comment on these JIRA issues or pull request. Once you get the architectural understanding, you may be helpful in reviewing other contributor patches. Feel free to provide your feedbacks on other contributor patches on pull request or in JIRA.

These comments can be pointing out issues, questions, style issues, typos etc. Involving in reviews will help better understanding various code paths, process of code contributions, and help more bonding with other contributions.

Not only code logic changes, you can also focus on following type of issues:

- Whether enough test cases added for the code change
- Whether code fully compliant to the Apache Mnemonic coding guidelines
- Make sure code changes are completely relevant to the current issue
- Check unused imports left out
- Make sure better javadocs/documentation/comments added
- Make sure better logging added
- Check the typos
- Ask user confirmation whether he tested manually some special cases if we can not automat some corner cases.
- Check for better class/method/variable names

Code Changes

Generally for code changes, contributor should have gained enough knowledge on the code base to change the code. Documentation would be the best first source for getting better understanding the system.

For code changes, contributors need to clone the repository to local machines. Since we follow the contributions via the git pull request method we need to follow certain steps.

- **Git Pull Request work flow**

  1. First of all, contributor needs github account if you don't have one already.
  2. Then fork the Apache Mnemonic git repository to your account.
  3. Then go to your machine's git shell command prompt and clone your forked repository. Command: git clone <FORKED_REPO.git>.

     This will have one remote that will be named as origin by default.
  4. Then git remote add original branch as upstream. Command: git remote add upstream <ORIGINAL_REPO.git>.

     You can update branch like :git fetch upstream/master

     So, now we have 2 remotes(origin, upstream) added.

     Origin is for forked repository and upstream is for original Mnemonic apache repository.
  5. Create local branch for your work at forked branch master. Command: `git checkout -b <branch-name>.`

     ```
     So, here master is default trunk code base branch and <branch-name> is

     the working branch on forked repo. Also you need to push this branch to forked repo.

     Let's do git push origin <branch-name>
     ```

6. Make sure you are on <branch-name> for working on your changes. Command: git checkout <branch-name>

   Then do your changes here.

7. Now you should commit changes to your <branch-name> and push the changes to forked repo.

8. Once you pushed your commit, create the pull request. To do that, go to the UI and login to your git account.

   Here you should see Create Pull Request option in UI. Just click on it and compare against original repo code base.

   Copy the pull request link from browser URL bar.

9. Add this pull request link into the corresponding JIRA issue created.

10. Once a committer review your pull request, he may +1 on it and merge to original repo. Or he may reject the changes.

    In rejected case, you may need to generate the pull request again after correcting the comments.

11. After pull request merged to main repo, you can safely remove this test branch.

12. Your forked master branch and original repo might differ due to other committer commits on original repo. So, we can keep rebase the forked repo master branch with original repo.

    Command: git rebase upstream/master

13. For any new issue you wanted to work on you can follow the same steps.

    You can name your working branch (referred as <branch-name> above) as with your JIRA issue id.

- **Coding guidelines while editing the code**

  Rules for Java code:
  1. Standard Java code conventions
  2. We follow 2-space indentation
  3. No TABs are allowed in the code
  4. LF newline
  5. Encode UTF-8
  6. Any new file addition must have Apache License header.
  7. Maximum length of line is 120 characters

  Rules for C/C++ code:
  1. Customized K&R
  2. We follow 2-space indentation
  3. No TABs are allowed in the code
  4. LF newline
  5. Encode UTF-8
  6. Any new file addition must have Apache License header.
  7. Maximum length of line is 120 characters

Helping Users/Developers On Mailing Lists

While using the Apache Mnemonic, users may get some questions/issues. We have mailing lists to discuss such questions or issues. After the discussions, if community concluded that as issue on code base, you can raise a JIRA ticket and track. If they are purely questions, then you may get replies from other users or developers.

To get involved in mailing lists, here you need to subscribe: dev , commits

Helping On Testing Release Candidates

All users/developers are encouraged to register to dev mailing list for getting announcement or vote about release candidates. User/contributors can help to test releases with their use cases and workloads. If you felt happy with the release candidate after testing, you can leave the feedback in mail list. You can also report issues if you find any while testing.